Builder and Dependency Injection

Builder:

The Builder Pattern is a design pattern used to construct complex objects step by step, allowing you to create different representations of an object using the same process.

public class Pizza {

public string Crust { get; set; }

public bool Cheese { get; set; }

public bool Pepperoni { get; set; }

}

public class PizzaBuilder {

private Pizza pizza = new Pizza();

public PizzaBuilder AddCrust(string crust) {

pizza.Crust = crust;

return this;

}

public PizzaBuilder AddCheese() {

pizza.Cheese = true;

return this;

}

public PizzaBuilder AddPepperoni() {

pizza.Pepperoni = true;

return this;

}

public Pizza Build() {

return pizza;

}

}

Public static void Main() {

var customPizza = new PizzaBuilder()

.AddCrust("Thin")

.AddCheese()

.AddPepperoni()

.Build();

}

Dependency Injection:

Dependency Injection is when you provide the dependencies (objects/services) a class needs from the outside, instead of creating them inside the class.

Example:

public interface ICar {

void Drive();

}

public class Tesla : ICar {

public void Drive()

{

Console.WriteLine("Driving a Tesla.");

}

}

public class Driver {

private ICar \_car;

public Driver(ICar car) {

\_car = car;

}

public void StartJourney() {

\_car.Drive();

}

}

Public static void Main() {

ICar myCar = new Tesla();

Driver driver = new Driver(myCar);

driver.StartJourney();

}